Statistical Process Control for Supporting OS-level Failure Prediction *

João R. Campos[1], **Rodrigo Pato Nogueira** [1]

[1]University of Coimbra, CISUC, DEI, Coimbra, Portugal

jrcampos@dei.uc.pt,pnogueira@student.dei.uc.pt

Abstract. *Software systems are used to execute critical tasks on a daily basis. Failures can easily lead to significant losses or even loss of lives. Online Failure Prediction (OFP) tries to predict incoming failures using the current state of the system. This relies on the premise that there are symptoms (i.e., some misbehavior of the system) prior to failure, however, characterizing the (mis)behavior of a complex system is an open issue. How can we know if the failure predictors are actually modeling the symptoms, and not just identifying correlations in the data? In this work, we explore the use of Statistical Process Control (SPC) to characterize the stability and instability of the Linux Operating System (OS).*

1. Introduction

Nowadays, software systems are used to execute critical tasks on a daily basis. Failures in such systems can easily lead to significant losses or even loss of lives. Facebook outage in 2019 cost approximately $90 million [Brown 2020] and software faults have been identified as a major contributor to Boeing 737 Max crashes [McFall-Johnsen 2020]. Online Failure Prediction (OFP) is a fault-tolerance technique that tries to predict incoming failures by using past data and the current state of the system [Salfner et al. 2010].

OFP has been successfully used in domains that rely on a specific set of metrics that can accurately characterize the state of the system (e.g., disk failure [Zhang et al. 2020]). However, OS-level failure prediction, whose characterization relies on hundreds of system metrics, is not trivial. Still, recent works (e.g., [Campos et al. 2023]) have shown that it is possible to create accurate OS-level failure predictors for various failure modes and OSs. These approaches focus on symptom monitoring, and they rely on the premise that the system will exhibit some out-of-norm behavior before a failure event. Given the plethora of different failures that can be observed in a complex system, determining whether or not there are in fact symptoms to monitor is still an open issue. The impact is clear: given the ability that Machine Learning (ML) algorithms have to detect patterns in the data, how can we know if there is, in fact, some symptom to monitor, or if the algorithms just leverage some correlation in the data?

Statistical Process Monitoring (SPM), or SPC, is a technique that is extensively used in industrial domains. SPC is a statistical tool for checking the conformance of products and systems to their operational requirements [Qiu 2013]. It tries to distinguish be-

*Work partially funded by Project No. 7059 - Neuraspace - AI fights Space Debris, reference C644877546-00000020, supported by the RRP - Recovery and Resilience Plan and the European Next Generation EU Funds, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation

tween *common cause* variation (i.e., variability inherent to the system) and *special cause* variation (i.e., variation caused by the malfunction of a component in the process).

Given the successful uses of SPC in industry, our main research question is: is it possible to apply the same principles to monitor modern complex digital systems? To explore this hypothesis, we conducted a thorough analysis of the state of the art in SPC techniques and applied them to a Linux failure dataset [Campos et al. 2022]. Taking the characteristics of the dataset into account, discussed further in *Section 3*, the technique selected for the analysis was PCA-MSPM. Using this technique, for some experiments it was possible to clearly observe the system becoming unstable up until the subsquent failure. This demonstrates the potential of such techniques, as they can be used for situations where the system might fail but existing detection mechanisms do not trigger an alert, as well as for forensics analysis of past failures.

2. Background and Related Work

OFP is a fault tolerance technique that tries to predict failures in the near future, using the current state of the system. It has been extensively used in domains such as disk failure (e.g., [Zhang et al. 2020]) and job failures in cloud applications (e.g., [Jassas and Mahmoud 2018]) where it is possible to define a set of metrics that accurately characterize the system. OS-level failure prediction is not so trivial as it requires hundreds of system metrics. Notwithstanding, recent works (e.g., [Campos et al. 2023]) have shown that with a proper methodology it is possible to create accurate failure predictors. The most promising OFP approaches focus on symptom monitoring. This means that they rely on the premise that the system will exhibit some out-of-norm behavior before a failure event. However, given the plethora of different failures that can be observed in a complex system, some of which just simply occur because some corruption occurred and immediately led to a failure, determining whether or not there are symptoms to monitor is still an open issue.

SPM, or SPC, is a technique that has been around since 1924 and that is extensively used in various domains such as industry and chemical engineering (e.g., [Reis et al. 2021]). Given the impact that a malfunctioning system can have, quality assurance and quality management are of utmost importance. SPC is a statistical tool for checking the conformance of products and systems to their specified requirements [Qiu 2013]. SPC tries to distinguish between variation inherent to the production system and variation that is caused by the malfunction of a component. These variations are related to the *stability/instability* of the system. In short, the in-control (stable) distribution of the data is analyzed and both Upper Control Limit (UCL) and Lower Control Limit (LCL) are computed, which define the Normal Operating Conditions (NOC) of the system. At runtime, the state of the system is analyzed and if it falls out of these ranges the system is no longer stable. Associated with this process, control charts are typically used to assist in the assessment. Given that modern systems are now typically characterized through multiple indicators, various techniques were developed for Multivariate Statistical Process Monitoring (MSPM). Other techniques such as Principal Component Analysis (PCA) were also proposed to address the high-dimensionality of modern systems. Additionally, techniques such as CUSUM, or EWMA have also been proposed to take into account the sequential evolution of the data [Qiu 2013]. Overall, the concept of SPC is identical to that of anomaly detection, however, these are supported by a statistical

background that allows identifying the NOC and its limits.

SPC techniques have been used for the task of failure prediction, but not in complex computer systems. As an example, [Ogden et al. 2017] used SPC to detect jet engine failures prior while [Liu et al. 2021] predicts failures of aero-engine load-bearing transmission system. [Lim et al. 2017] develop an SPC model for predicting the failure of a printed circuit board. Recent works have also tried to combine SPC with ML techniques [Tran 2022].

3. Experimental Methodology and Preliminary Results

The dataset used for this analysis is available at [Campos et al. 2022]. Briefly, it is a comprehensive failure dataset that was generated using fault injection on the Linux OS. Various types of faults were used in the experimental process along with various workloads to exercise the system. *Golden runs* (where no fault was injected) were also executed to establish the baseline behavior of the system. The subset of the data used for this task is comprised of 371 system metrics, all numeric, collected every second. The dataset comprises various failure modes. This dataset has been studied in various works, which have shown that it is possible to create accurate failure predictors for the various failure modes [Campos et al. 2023]. However, given the number of features and the complexity of the various failure modes, it is not yet clear when the symptoms start to occur.

To explore whether SPC can be used to assist in this task, because we are working with multivariate data, with high dimensionality and correlation, the PCA-MSPM process using Hotelling T^2 statistics was chosen. SPC processes are typically divided into two phases. *Phase I* aims to assess the stability of the system, by analyzing historic data from the process (the golden runs) to define preliminary control limits and checking if the monitoring statistics fall inside the NOC region (i.e., within the control limits). In *Phase II* we use these limits to determine if the system is in or out of control. It is important to notice that control limits result from the inherent variability of the process. PCA is used for two main reasons: *i)* to reduce the dimensionality of the data while maximizing explained variance in the data; *ii)* and to remove correlation in the data (Hotelling T^2 cannot handle high correlation). To determine the number of components to keep, the elbow method was used, as illustrated in *Figure 1*. 25 components were selected, which account for 70% of the variation in the data. The LCL and UCL were defined based on the golden runs data projected through PCA.

Figure 2 presents the control chart for Phase I T^2 (the x axis are the golden runs samples, the y the corresponding T^2 statistic). The red line identifies the UCL, above which it is considered that the system is out of control. As can be observed, almost every sample is below the UCL, which means the system is under NOC. Although there is variation in the data, this is modulated as variation inherent to the process.

In Phase II we assessed the stability of the failure experiments. The initial results were rather interesting, as for some failures the system exhibited a growing instability, as can be seen in *Figure 3*. In this particular experiment, using this technique it is possible to observe that there are in fact symptoms of an incoming failure, as the system is steadily moving away from NOC. However, there were also experiments where the symptoms are not so clear. In *Figure 4*, which belongs to a *hang* failure, it is also possible to observe that the system has an uncommon behavior, but it is not always out of control.

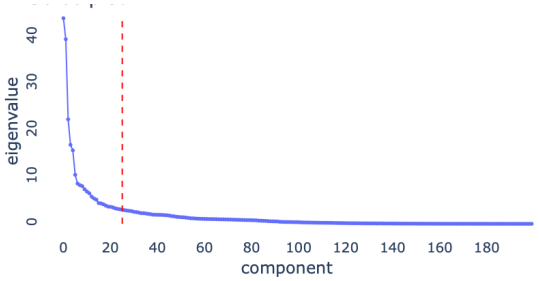

Figure 1. PCA

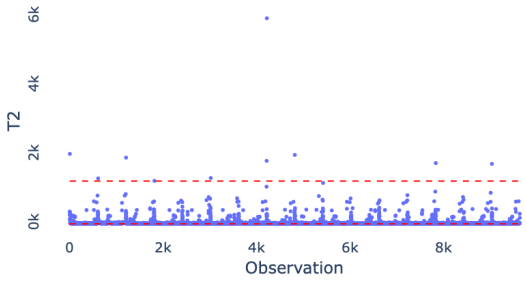

Figure 2. Phase 1 T^2 - Golden

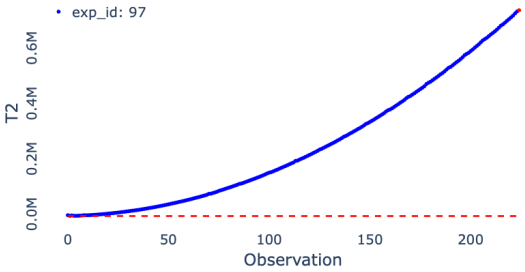

Figure 3. Phase 2 - CPU Failure

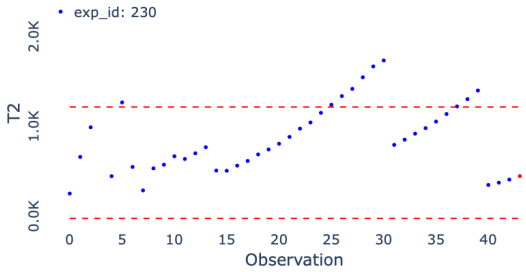

Figure 4. Phase 2 - Hang Failure

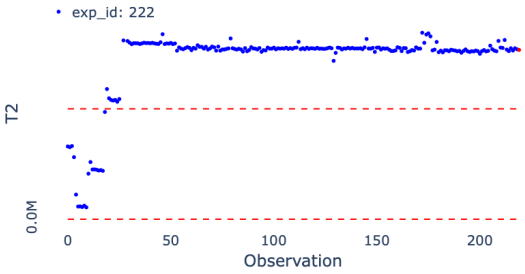

Figure 5. Phase 2 - CPU Failure

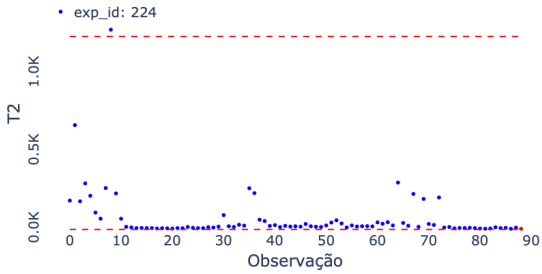

Figure 6. Phase 2 - Crash Failure

A similar observation can be made in *Figure 5* and *Figure 6*. In the first, the system moves out of control at the beginning of the experiment and stays there until a failure occurs later in the experiment. Although there are symptoms, they are not exactly progressively moving away from NOC. As for the latter, there is some variation at the beginning of the experiment, however, the system as a whole never gets out of control.

One of the possible reasons for these differences, besides the fact that there might not actually be symptoms, is that the best indicators for certain types of failures are features that have low variance throughout the golden runs. In fact, the feature with the highest importance for the crash failure mode is constant in every golden run. As a result, it has 0 contribution to the components obtained from PCA. This means that in Phase II, even strong variations in these features will not change the computed T^2 statistical value.

4. Conclusion

In this paper, we explored whether SPC could also be used for monitoring the stability of a modern OS. Results have shown promising results, making it possible to observe a system moving towards instability and subsequent failure. Notwithstanding, further research is necessary to generalize the process to other types of failures and causes.

5. References

References

Brown, B. (2020). Facebook's catastrophic blackout could cost $90 million in lost revenue. Accessed 2023-05-24.

Campos, J. R., Costa, E., and Vieira, M. (2022). A dataset of linux failure data for dependability evaluation and improvement. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 88–95. IEEE.

Campos, J. R., Costa, E., and Vieira, M. (2023). Online failure prediction through fault injection and machine learning: Methodology and case study. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 451–461. IEEE.

Jassas, M. and Mahmoud, Q. H. (2018). Failure analysis and characterization of scheduling jobs in google cluster trace. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3102–3107. IEEE.

Lim, H. K., Kim, Y., and Kim, M.-K. (2017). Failure prediction using sequential pattern mining in the wire bonding process. *IEEE Transactions on Semiconductor Manufacturing*, 30(3):285–292.

Liu, J., Pan, C., Lei, F., Hu, D., and Zuo, H. (2021). Fault prediction of bearings based on lstm and statistical process analysis. *Reliability Engineering & System Safety*, 214:107646.

McFall-Johnsen, M. (2020). Catastrophic software errors doomed boeing's airplanes. Accessed 2023-05-01.

Ogden, D. A., Arnold, T. L., and Downing, W. D. (2017). A multivariate statistical approach for anomaly detection and condition based maintenance in complex systems. In *2017 IEEE AUTOTESTCON*, pages 1–8. IEEE.

Qiu, P. (2013). *Introduction to statistical process control*. CRC press.

Reis, M. S., Rendall, R., Rato, T. J., Martins, C., and Delgado, P. (2021). Improving the sensitivity of statistical process monitoring of manifolds embedded in high-dimensional spaces: The truncated-q statistic. *Chemometrics and Intelligent Laboratory Systems*, 215:104369.

Salfner, F., Lenk, M., and Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys (CSUR)*, 42(3):10:1–10:42.

Tran, K. P. (2022). Introduction to control charts and machine learning for anomaly detection in manufacturing. *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*, pages 1–6.

Zhang, J., Zhou, K., Huang, P., He, X., Xie, M., Cheng, B., Ji, Y., and hu Wang, Y. (2020). Minority disk failure prediction based on transfer learning in large data centers of heterogeneous disk systems. *IEEE Transactions on Parallel and Distributed Systems*.